iControl® REST User Guide

Version 11.5

Table of Contents

Legal Notices

Publication Date

This document was published on January 29, 2014.

Publication Number

MAN-0525-00

Copyright

Trademarks

Patents

This product may be protected by one or more patents indicated at:
http://www.f5.com/about/guidelines-policies/patents

Export Regulation Notice

This product may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this product from the United States.

RF Interference Warning

This is a Class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

FCC Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This unit generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

Canadian Regulatory Compliance

This Class A digital apparatus complies with Canadian ICES-003.

Standards Compliance

This product conforms to the IEC, European Union, ANSI/UL and Canadian CSA standards applicable to Information Technology products at the time of manufacture.

Acknowledgments

Acknowledgments

This product includes Intel QuickAssist kernel module, library, and headers software licensed under the GNU General Public License (GPL).

This product includes gd-libgd library software developed by the following in accordance with the following copyrights:

- Portions copyright ©1994, 1995, 1996, 1997, 1998, 2000, 2001, 2002 by Cold Spring Harbor Laboratory. Funded under Grant P41-RR02188 by the National Institutes of Health.
- Portions copyright ©1996, 1997, 1998, 1999, 2000, 2001, 2002 by Boutell.Com, Inc.
- Portions relating to GD2 format copyright ©1999, 2000, 2001, 2002 Philip Warner.
- Portions relating to PNG copyright ©1999, 2000, 2001, 2002 Greg Roelofs.
- Portions relating to gdttf.c copyright ©1999, 2000, 2001, 2002 John Ellson (ellson@lucent.com).
- Portions relating to gdft.c copyright ©2001, 2002 John Ellson (ellson@lucent.com).
- Portions copyright ©2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007 2008 Pierre-Alain Joye (pierre@libgd.org).
- Portions relating to JPEG and to color quantization copyright ©2000, 2001, 2002, Doug Becker and copyright ©1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, Thomas G. Lane. This software is based in part on the work of the Independent JPEG Group.
- Portions relating to WBMP copyright 2000, 2001, 2002 Maurice Szmurlo and Johan Van den Brande. Permission has been granted to copy, distribute and modify gd in any context without fee, including a commercial application, provided that this notice is present in user-accessible supporting documentation.

This product includes software developed by Oracle America, Inc. Copyright ©2012.

1. Java Technology Restrictions. Licensee shall not create, modify, change the behavior of, or authorize licensees of licensee to create, modify, or change the behavior of, classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Oracle in any naming convention designation. In the event that Licensee creates an additional API(s) which: (a) extends the functionality of a Java Environment; and (b) is exposed to third party software developers for the purpose of developing additional software which invokes such additional API, Licensee must promptly publish broadly an accurate specification for such API for free use by all developer.
2. Trademarks and Logos. This License does not authorize an end user licensee to use any Oracle America, Inc. name, trademark, service mark, logo or icon. The end user licensee acknowledges that Oracle owns the Java trademark and all Java-related trademarks, logos and icon including the Coffee Cup and Duke ("Java Marks") and agrees to: (a) comply with the Java Trademark Guidelines at http://www.oraclc.com/html/3party.html; (b) not do anything harmful to or inconsistent with Oracle's rights in the Java Marks; and (c) assist Oracle in protecting those rights, including assigning to Oracle any rights acquired by Licensee in any Java Mark.
3. Source Code. Software may contain source code that, unless expressly licensed for other purposes, is provided solely for reference purposes pursuant to the terms of your license. Source code may not be redistributed unless expressly provided for in the terms of your license.
4. Third Party Code. Additional copyright notices and license terms applicable to portion of the Software are set forth in the THIRDPARTYLICENSEREADME.txt file.
5. Commercial Features. Use of the Commercial Features for any commercial or production purpose requires a separate license from Oracle. "Commercial Features" means those features identified in Table I-I (Commercial Features In Java SE Product Editions) of tile Software documentation accessible at http://www.oracle.com/technetwork/java/javase/documentation/index.html.

This product includes utilities developed by Linus Torvalds for inspecting devices connected to a USB bus.

This product includes perl-PHP-Serialization software, developed by Jesse Brown, copyright ©2003, and distributed under the Perl Development Artistic License (http://dev.perl.org/licenses/artistic.html).

This product includes software developed by members of the CentOS Project under the GNU Public License, copyright ©2004-2011 by the CentOS Project.

4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes software licensed from William Ferrell, Selene Scriven and many other contributors under the GNU General Public License, copyright ©1998 - 2006.

This product includes software developed by Thomas Williams and Colin Kelley. Copyright ©1986 - 1993, 1998, 2004, 2007

Permission to use, copy, and distribute this software and its documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Permission to modify the software is granted, but not the right to distribute the complete modified source code. Modifications are to be distributed as patches to the released version. Permission to distribute binaries produced by compiling modified sources is granted, provided you

1. distribute the corresponding source modifications from the released version in the form of a patch file along with the binaries,
2. add special version identification to distinguish your version in addition to the base release version number,
3. provide your name and address as the primary contact for the support of your modified version, and
4. retain our contact information in regard to use of the base software.

Permission to distribute the released version of the source code along with corresponding source modifications in the form of a patch file is granted with same provisions 2 through 4 for binary distributions. This software is provided "as is" without express or implied warranty to the extent permitted by applicable law.

This product includes software developed by the Computer Systems Engineering Group at Lawrence Berkeley Laboratory. Copyright ©1990-1994 Regents of the University of California. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the Computer Systems Engineering Group at Lawrence Berkeley Laboratory.

This product includes the ixgbevf Intel Gigabit Linux driver, Copyright © 1999 - 2012 Intel Corporation, and distributed under the GPLv2 license, as published by the Free Software Foundation.

This product includes libwebp software. Copyright © 2010, Google Inc. All rights reserved.

This product includes Angular software developed by Google, Inc., http://angulargs.org, copyright © 2010-2012 Google, Inc., and distributed under the MIT license.

This product includes node.js software, copyright © Joyent, Inc. and other Node contributors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

• The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter

1

REST

- *About Representational State Transfer*
- *About URI format*
- *About HTTP method semantics*
- *About JSON format*

About Representational State Transfer

Representational State Transfer (REST) describes an architectural style of web services where clients and servers exchange representations of resources. The REST model defines a resource as a source of information, and also defines a representation as the data that describes the state of a resource. REST web services use the HTTP protocol to communicate between a client and a server, specifically by means of the POST, GET, PUT, and DELETE methods, to create, read, update, and delete elements or collections. In general terms, REST queries resources for the configuration objects of a BIG-IP® system, and creates or modifies the representations of those configuration objects.

The iControl® REST implementation follows the REST model by:

- Using REST as a resource-based interface, and creating API methods based on nouns.
- Employing a stateless protocol and MIME data types, as well as taking advantage of the authentication mechanisms and caching built into the HTTP protocol.
- Supporting the JSON format for document encoding.
- Representing the hierarchy of resources and collections with a Uniform Resource Identifier (URI) structure.
- Returning HTTP response codes to indicate success or failure of an operation.
- Including links in resource references to accommodate discovery.

About URI format

The iControl® REST API enables the management of a BIG-IP® device by using web service requests. A principle of the REST architecture describes the identification of a resource by means of a Uniform Resource Identifier (URI). You can specify a URI with a web service request to create, read, update, or delete some component or module of a BIG-IP system configuration. In the context of REST architecture, the system configuration is the representation of a resource. A URI identifies the name of a web resource; in this case, the URI also represents the tree structure of modules and components in `tmsh`.

In iControl REST, the URI structure for all requests includes the string `/mgmt/tm/` to identify the namespace for traffic management. Any identifiers that follow the endpoint are considered to be resource collections.

Tip: *Use the default administrative account, `admin`, for requests to iControl REST. Once you are familiar with the API, you can create user accounts for iControl REST users with various permissions.*

```
https://management-ip/mgmt/tm/module
```

The URI in the previous example designates all of the `tmsh` subordinate modules and components in the specified `module`. iControl REST refers to this entity as an *organizing collection*. An organizing collection contains links to other resources. The `management-ip` component of the URI is the fully qualified domain name (FQDN) or IP address of a BIG-IP device.

Attention: *iControl REST only supports secure access through HTTPS, so you must include credentials with each REST call. Use the same credentials you use for the BIG-IP device manager interface.*

For example, use the following URI to access all of the components and subordinate modules in the `ltm` module:

```
https://192.168.25.42/mgmt/tm/ltm
```

The URI in the following example designates all of the subordinate modules and components in the specified *sub-module*. iControl REST refers to this entity as a collection; a collection contains resources.

`https://`*`management-ip`*`/mgmt/tm/`*`module`*`/`*`sub-module`*

The URI in the following example designates the details of the specified *`component`*. The *Traffic Management Shell (tmsh) Reference* documents the hierarchy of modules and components, and identifies details of each component. iControl REST refers to this entity as a resource. A resource may contains links to sub-collections.

`https://`*`management-ip`*`/mgmt/tm/`*`module`*`[/`*`sub-module`*`]/`*`component`*

About reserved ASCII characters

To accommodate the BIG-IP® configuration objects that use characters, which are not part of the unreserved ASCII character set, use a percent sign (%) and two hexadecimal digits to represent them in a URI. The unreserved character set consists of: `[A - Z] [a - z] [0 - 9] dash (-), underscore (_), period (.), and tilde(~)`

You must encode any characters that are not part of the unreserved character set for inclusion in a URI scheme. For example, an IP address in a non-default route domain that contains a percent sign to indicate an address in a specific route domain, such as `192.168.25.90%3`, should be encoded to replace the `%` character with `%25` .

About REST resource identifiers

A URI is the representation of a resource that consists of a protocol, an address, and a path structure to identify a resource and optional query parameters. Because the representation of folder and partition names in `tmsh` often includes a forward slash (/), URI encoding of folder and partition names must use a different character to represent a forward slash in iControl® REST. To accommodate the forward slash in a resource name, iControl REST maps the forward slash to a tilde (~) character. When a resource name includes a forward slash (/) in its name, substitute a tilde (~) for the forward slash in the path. For example, a resource name, such as `/Common/plist1`, should be modified to the format shown here:
`https://management-ip/mgmt/tm/security/firewall/port-list/~Common~plist1`

About HTTP method semantics

Hypertext Transfer Protocol (HTTP 1.1) describes the functionality of HTTP methods and the role of the Uniform Resource Identifier (URI) that identifies a collection or resource. The composition of a URI includes endpoints, collections, resources, and options. Endpoints are elements of a namespace, such as `mgmt`, which define the path to a resource or collection. A collection is a set or resources of the same type, and in iControl® REST, a collection is a collection of resources or an organizing collection of links to resources. Finally, options are query parameters that refine the result set from a request. In the context of an HTTP method, a URI identifies a resource or collection as the target of an operation.

The URI that identifies the namespace for a request also affects the actions of a method. For a POST request, a URI indicates a resource under which to create a subordinate resource. The subordinate resource is considered a new entity, and not a modification of an existing entity. If the resource exists, the protocol considers a request to create the same resource as an error. For a PUT request, if the URI refers to an existing resource, the entity is considered to be a modification of an existing resource.

iControl REST has semantics that behave differently for collections and resources. The semantics of the HTTP methods in iControl REST are described in this table.

Method	Description
GET	For both collections and resources, iControl REST supports the GET operation.
POST	For both collections and resources, iControl REST supports the POST operation.
DELETE	For collections, iControl REST does not support the DELETE operation. For resources, iControl REST supports the DELETE operation.
PUT	For collections, iControl REST does not support the PUT operation. For resources, iControl REST partially supports the PUT operation.
PATCH	For collections, iControl REST does not support the PATCH operation. For resources, iControl REST supports the PATCH operation.

About JSON format

iControl® REST formats a response to a request in JavaScript Object Notation (JSON) format. When iControl REST processes a GET request, it generates a response code and a text body in JSON format. To indicate the format of the text body in a response, the HTTP `Content-Type` header indicates the content type as `application/json`. Likewise, an error response contains additional descriptive text in JSON format. A response from iControl REST contains properties. Properties can describe a configuration object, or the statistics for a resource. In iControl REST, the term *property* refers to a name/value, or key/value, pair in a JSON object. For a GET request, the properties consist of JSON objects or arrays, or both. JavaScript Object Notation (JSON) defines a data interchange format that facilitates communication between clients and servers. Similar to the eXtensible Markup Language (XML) common to SOAP web services, JSON describes a structuring of data for exchange between clients and servers in REST web service requests.

The JSON terminology consists of two structures: objects and arrays. An object is a collection of one or more name/value pairs, as shown:

```
{ "partition":"Common" }
```

Both the name and value appear in double quotes (" "), and a colon (:) separates the name and the value in the string. For objects that contain multiple name pairs, a comma (,) separates additional name/value pairs. A JSON value must be an object, array, number, string, or one of three literal names: `false`, `null`, or `true`. The other structure for a property is a JSON array, which is an ordered list of values, as shown:[{ "components":8, "security":"disabled" }]In JSON structure, square brackets ([ ]) enclose the objects in an array. Any objects between the square brackets follow the JSON standard for name/value pairs. Collectively, the name/value pairs are the properties of a BIG-IP® system configuration. For iControl REST, the name/value pairs can be thought of as property name and property value.

Many of the examples in this guide use `curl`, a common `Unix` utility for accessing URIs, as shown in the example to make an iControl REST call on a particular BIG-IP device. The `curl` utility uses the following general syntax:

```
curl -k -u username:password -X http-method uri
```

Where:

* *username* and *password* specify the credentials of a BIG-IP system administrator, such as `admin:admin`;
* *http-method* specifies a verb, such as GET, POST, PUT, or DELETE ; and
* *uri* identifies the BIG-IP system resource (for example, `192.168.25.42/mgmt/tm/ltm`).

A response from iControl REST shows the properties and reference links as JSON objects and arrays.

```
{
    "kind":"tm:ltm:ltmcollectionstate",
    "selfLink":"https://localhost/mgmt/tm/ltm?ver=11.5.0",
    "items":[
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/auth?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/classification?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/data-group?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/dns?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/global-settings?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/html-rule?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/message-routing?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/monitor?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/profile?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/default-node-monitor?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/ifile?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/lsn-pool?ver=11.5.0"
```

```
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/nat?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/node?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/policy?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/policy-strategy?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/pool?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/rule?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/snat?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/snat-translation?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/snatpool?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/traffic-class?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/virtual?ver=11.5.0"
            }
         },
         {
            "reference":{
               "link":"https://../mgmt/tm/ltm/virtual-address?ver=11.5.0"
            }
         }
      ]
}
```

About additional properties

The iControl® REST implementation includes some document properties not present in *Traffic Management Shell (tmsh) Reference* output. The differences appear in a response to a GET request of a collection or resource, as shown in the example.

PropertyName	Description
kind	A unique type identifier.
generation	A generation number for a resource. Modification of a resource, or a related resource, changes the value. The value does not necessarily increase monotonically. For example, if you modify a resource in a sub-collection, the modification may cause a change in the parent object.
selfLink	A link to this resource.

```
{
    "kind":"tm:sys:software:image:imagecollectionstate",
    "selfLink":"https://localhost/mgmt/tm/sys/software/image?ver=11.5.0",
    "items":[
        {
            "kind":"tm:sys:software:image:imagestate",
            "name":"BIGIP-11.5.0.0.0.191.iso",
            "fullPath":"BIGIP-11.5.0.0.0.191.iso",
            "generation":38,

"selfLink":"https://../mgmt/tm/sys/software/image/BIGIP-11.5.0.0.0.191.iso?ver=11.5.0",

            "build":"0.0.191",
            "buildDate":"Wed Nov 27 14 03 09 PST 2013",
            "checksum":"fab5b673486ccc1ec20fbe6cea51df50",
            "fileSize":"1751 MB",
            "lastModified":"Tue Dec  3 01:30:32 2013",
            "product":"BIG-IP",
            "verified":"yes",
            "version":"11.5.0"
        },
        {
            "kind":"tm:sys:software:image:imagestate",
            "name":"BIGIP-tmos-bugs-staging-11.5.0.0.0.237.iso",
            "fullPath":"BIGIP-tmos-bugs-staging-11.5.0.0.0.237.iso",
            "generation":37,

"selfLink":"https://../software/image/BIGIP-tmos-bugs-staging-11.5.0.0.0.237.iso?ver=11.5.0",

            "build":"0.0.237",
            "buildDate":"Wed Dec 4 14 14 44 PST 2013",
            "checksum":"bb4ae4838a5743fa209f67a1b56dedef",
            "fileSize":"1843 MB",
            "lastModified":"Wed Dec  4 15:32:28 2013",
            "product":"BIG-IP",
            "verified":"yes",
            "version":"11.5.0"
        }
    ]
}
```

```
root@(BIG-IP1)(...)(tmos)# list sys software image
sys software image BIGIP-11.4.0.321.0.iso {
    build 321.0
    build-date "Mon Feb 11 07 23 24 PST 2013"
```

```
    checksum f9411fde01d6a3521d4ae393e9bb077c
    file-size "1522 MB"
    last-modified "Mon Feb 11 09:35:50 2013"
    product BIG-IP
    verified yes
    version 11.4.0
}
root@(BIG-IP1)(...)(tmos)#
```

About null values and properties

Flags are typically composed as a bit set by software to indicate state, such as 0 or 1, and indicate on or off, respectively. iControl® REST displays flags that are set with the flag name and a value of null. If the value of a flag is none, iControl REST omits the property from the output.

Note:

To POST or PUT a flag with only a single value, enter the property name in the JSON body with a value of null.

```
{
    "kind":"tm:sys:software:volume:volumecollectionstate",
    "selfLink":"https://localhost/mgmt/tm/sys/software/volume?ver=11.5.0",
    "items":[
        {
            "kind":"tm:sys:software:volume:volumestate",
            "name":"MD1.1",
            "fullPath":"MD1.1",
            "generation":34,

"selfLink":"https://localhost/mgmt/tm/sys/software/volume/MD1.1?ver=11.5.0",
            "basebuild":"0.0.191",
            "build":"0.0.191",
            "product":"BIG-IP",
            "status":"complete",
            "version":"11.5.0",
            "media":[
                {
                    "name":"MD1.1",
                    "media":"array",
                    "size":"default"
                }
            ]
        },
        {
            "kind":"tm:sys:software:volume:volumestate",
            "name":"MD1.2",
            "fullPath":"MD1.2",
            "generation":35,

"selfLink":"https://localhost/mgmt/tm/sys/software/volume/MD1.2?ver=11.5.0",
            "active":null,
            "apiRawValues":{

            },
            "basebuild":"0.0.237",
            "build":"0.0.237",
            "product":"BIG-IP",
            "status":"complete",
            "version":"11.5.0",
            "media":[
                {
```

```
                "name":"MD1.2",
                "defaultBootLocation":null,
                "media":"array",
                "size":"default"
            }
        ]
    },
    {
        "kind":"tm:sys:software:volume:volumestate",
        "name":"MD1.3",
        "fullPath":"MD1.3",
        "generation":36,

"selfLink":"https://localhost/mgmt/tm/sys/software/volume/MD1.3?ver=11.5.0",
        "status":"complete",
        "media":[
            {
                "name":"MD1.3",
                "media":"array",
                "size":"default"
            }
        ]
    }
  ]
}
```

About reserved property names

iControl® REST reserves several property names, most notably, the words `name` and `generation`. Some `tmsh` components include properties with reserved property names. When iControl REST encounters a reserved name in the JSON body, it replaces the reserved names with the corresponding replacement, `tmName` or `tmGeneration`.

About property name format differences

Property and option names in iControl® REST use a different naming convention than *Traffic Management (tmsh) Shell*. In `tmsh`, property names consist of lowercase characters. For property names that contain multiple words, hyphens separate the words. iControl REST uses camel case convention for property names, where the first word of a property is lowercase, and all additional words in the name are capitalized.

For example, the property `build-date`, as shown in `tmsh`, appears as `buildDate` in iControl REST.

Chapter

2

GET requests

- *Discovering modules and components*

Discovering modules and components

The top-level modules in tmsh form the basis of discovery for components in tmsh. iControl® REST supports a subset of more than 600 components that exist in tmsh. The *Traffic Management Shell (tmsh) Reference* lists all tmsh modules and components, most of which appear under the same names in tmsh.

The tmsh root modules are:

- actions
- analytics
- apm
- asm
- auth
- cli
- cm
- gtm
- ltm
- net
- pem
- security
- sys
- transaction
- util
- vcmp
- wam
- wom

If you are familiar with command-line tools, use curl, or a similar utility, to make a request to iControl REST and specify a organizing collection. For example, the command: curl -k -u admin:admin -X GET https://192.168.25.42/mgmt/tm/ltm makes a request of the ltm organizing collection.

Note: The contents of an iControl REST resource may not have all of the properties and options of its tmsh counterpart below the sub-collection level.

To discover the structure, make a request to iControl REST with the GET method and specify an organizing collection, as shown in this example. If you expand the links in the response to this request, you can discover the structure of iControl REST.

```
GET https://192.168.25.42/mgmt/tm/ltm
```

```
{
    "items":[
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/ltm/auth?ver=11.5.0"
            }
        },
        {
            "reference":{
```

```
                   "link":"https://../mgmt/tm/ltm/classification?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/data-group?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/ltm/dns?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/global-settings?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/html-rule?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/message-routing?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/monitor?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/profile?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/default-node-monitor?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/ifile?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/lsn-pool?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/nat?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/node?ver=11.5.0"
            }
        },
```

```
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/policy?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/policy-strategy?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/pool?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/rule?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/snat?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/snat-translation?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/snatpool?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/traffic-class?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/virtual?ver=11.5.0"
        }
    },
    {
        "reference":{
            "link":"https://../mgmt/tm/ltm/virtual-address?ver=11.5.0"
        }
    }
],
"kind":"tm:ltm:ltmcollectionstate",
"selfLink":"https://localhost/mgmt/tm/ltm?ver=11.5.0"
}
```

Note: A module that is not provisioned on a BIG-IP® system will not appear in the output.

This example expands one of the links in the response from the previous request by making another GET request to iControl REST for a collection.

```
GET https://192.168.25.42/mgmt/tm/ltm/persistence
```

```
{
    "kind":"tm:ltm:persistence:persistencecollectionstate",
    "selfLink":"https://localhost/mgmt/tm/ltm/persistence?ver=11.5.0",
    "items":[
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/cookie?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/dest-addr?ver=11.5.0"
            }
        },
        {
            "reference":{

"link":"https://../mgmt/tm/ltm/persistence/global-settings?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/hash?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/msrdp?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/sip?ver=11.5.0"
            }
        },
        {
            "reference":{
                 "link":"https://../mgmt/tm/ltm/persistence/source-addr?ver=11.5.0"

            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/ssl?ver=11.5.0"
            }
        },
        {
            "reference":{
                "link":"https://../mgmt/tm/ltm/persistence/universal?ver=11.5.0"
            }
        }
    ]
}
```

Note: Some JSON formatting tools employ different algorithms to sort names, which affects the order in which properties are displayed.

Make a GET request and specify one of the links in the response from the previous request. The response contains the properties of the sub-collection.

```
GET https://192.168.25.42/mgmt/tm/ltm/persistence/global-settings
```

```
{
    "kind":"tm:ltm:persistence:global-settings:global-settingsstate",
    "selfLink":"https://../mgmt/tm/ltm/persistence/global-settings?ver=11.5.0",

    "destAddrLimitMode":"timeout",
    "destAddrMax":2048,
    "proxyGroup":"/Common/aol"
}
```

About query parameters

Because collections can produce large result sets, iControl® REST implements a subset of the Open Data Protocol (OData) recommendations for query languages and system query options. The OData protocol defines query parameters for use in a URI to manage a result set. For example, you can include or exclude rows from a result set, constrain a query to resources contained within an administrative partition, or specify a particular version of iControl REST. The GET request is the HTTP method to retrieve the configuration of an object, and iControl REST supports GET requests against collections and resources. With one exception, query parameters are used in GET requests.

To use a query parameter, append a query parameter expression to the end of a request URI. All query parameter expressions begin with a question mark (?), followed by a query parameter name, a comparison or logical operator, and a value. A value adheres to the camel case naming convention for iControl REST. OData query parameter names also contain a dollar sign ($). To specify additional query parameters, precede the parameter with an ampersand (&), then specify the query parameter expression. For example, you can specify that the response only include the name property in the following request:

```
GET https://localhost/tm/ltm/pool/?$select=name
```

The following table lists the parameters that are iControl REST implementations of the OData query parameters. All OData query parameters being with a dollar sign ($). Note that the $filter parameter, if used, limits the result set to a specific administrative partition.

Parameter	Description
$filter	Specifies an administrative partition to query for a result set. This parameter filters the result set by partition name and does not fully implement the corresponding OData query parameter. The asm module fully implements the OData query parameter.
$select	Specifies a subset of the properties that will appear in the result set.
$skip	Specifies the number of rows to skip in the result set. The result set is chosen from the remaining rows.
$top	Specifies the first N rows of the result set.

iControl REST supports comparison and logical operators as described in the OData recommendation.

Operator	Description
eq	Equal to
ne	Not equal to
lt	Less than
le	Less than or equal to
gt	Greater than
ge	Greater than or equal to
and	True if both operands are true
or	True if either operand is true
not	Negation of operand

Note: *iControl REST supports only the* `eq` *operator with the $filter parameter.*

iControl REST includes several custom query parameters. The custom query parameters do not include a dollar sign ($) character in the parameter name.

Parameter	Description
expandSubcollections	Specifies that iControl REST expand any references to sub collections when set to `true`. By default, the response to a GET request only contains links for sub-collection reference properties.
options	Specifies the options to a query request. This parameter takes values that are compatible with the `tmsh` command-line options.
ver	Specifies the version number of the iControl® REST API to use when making a request. Defaults to the current version, if you do not specify a value.

About API versions

Over time, modifications to the iControl® REST API may require a new version number. To restrict requests and responses to a particular version of the API, iControl REST includes the API version parameter as an option to a URI. To use a particular API version, specify the `ver` parameter, an API version number, such as 11.5.0, and append the string to the end of the URI, just as you would with any query parameter.

```
GET https://192.168.25.42/mgmt/tm/ltm?ver=11.5.0
```

The JSON body for a response includes an API version number in the `selfLink` property, as well as any links. For iControl REST, the version number of a resource in a response matches the version number sent in a request. If you do not specify the version of the API, the version defaults to the current version. To maintain backward compatibility with future releases of the API, a response will contain resources that match the version number specified in the request. If iControl REST cannot generate a response that is compatible with the request, it returns an error code.

Note: *Although some REST implementations use HTTP headers to manage version information, iControl REST does not use any HTTP headers to identify an API.*

About paging properties

iControl® REST supports pagination options for large collections. The implementation of pagination utilizes the Open Data Protocol (OData) query parameters to provide information that you can use to navigate a large result set. When you request a large collection, the iControl® REST response include properties to identify the URI for the collection, the next page of the result set, the previous page of the result set, as well as the total number of items in the result, total number of pages, the current page, the number of items per page, and a count of the number of items in the current page. iControl® REST calculates these values on the filtered result set.

Property	Description
selfLink	The URI of the collection, including any query parameters.
nextLink	The next set of data in the result set. Includes the $skip query parameter in the link.
previousLink	The previous set of data in the result set. Not present in the first set of data.
currentItemCount	A count of the number of items in the result set, either as the value of the $top query parameter, or the remaining number of items if less than the number requested.
itemsPerPage	The number of items to display per page.
pageIndex	The current page in the result set.
totalPages	The total number of pages in the result set, equal to the result of (totalItems / itemsPerPage), rounded up to the next integer value.
startIndex	The index of the first item in the result set.
totalItems	The number of items in the result set, as calculated by the $inlinecount=allpages query parameter.

Paging through large collections

Collections that contain a large number of items consume a great deal of network bandwidth and processing power if processed in a single GET request. Query parameters allow you to manage multi page responses. iControl® REST supports the OData system query parameters $top and $skip to return pages items sets.

Use the $top query parameter to specify the maximum number of items for the BIG-IP® device to return. If you use curl and run this command from a Unix command line, precede the dollar sign character ($) with a backslash character (\) to prevent shell interpretation of the character.

```
curl -k -u admin:admin -X GET https://192.168.25.42/mgmt/tm/sys?\$top=4
```

To query for the first n data items, specify the URI, and append the $top query parameter to the URI. This query displays the first four items in the sys collection output. The response indicates the nextLink and previousLink properties that serve as navigation markers to the next page and previous page, respectively.

```
https://192.168.25.42/mgmt/tm/sys?$top=4
```

```
{ "currentItemCount" : 4,
  "items" : [
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/application?ver=11.5.0" } }
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/crypto?ver=11.5.0" } }
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/daemon-log-settings?ver=11.5.0" }
}
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/disk?ver=11.5.0" } }
  ],
  "itemsPerPage" : 4,
  "kind" : "tm:sys:syscollectionstate",
  "nextLink" : "https://localhost/mgmt/tm/sys?$top=4&$skip=4&ver=11.5.0",
  "pageIndex" : 1,
  "selfLink" : "https://localhost/mgmt/tm/sys?$top=4&ver=11.5.0",
  "startIndex" : 1,
  "totalItems" : 36,
  "totalPages" : 9
}
```

To request the next n data items, use the same URI as the previous example and append the $skip query parameter to the URI. This example displays the next four items in the sys collection output. The response also indicates the nextLink and previousLink properties that serve as navigation markers into the data.

```
https://192.168.25.42/mgmt/tm/sys?$top=4&$skip=4
```

```
{ "currentItemCount" : 4,
  "items" : [
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/file?ver=11.5.0" } },
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/icall?ver=11.5.0" } },
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/log-config?ver=11.5.0" } },
    { "reference" :
      { "link" : "https://../mgmt/tm/sys/sflow?ver=11.5.0" } }
  ],
  "itemsPerPage" : 4,
  "kind" : "tm:sys:syscollectionstate",
  "nextLink" : "https://localhost/mgmt/tm/sys?$top=4&$skip=8&ver=11.5.0",
  "pageIndex" : 2,
  "previousLink" : "https://localhost/mgmt/tm/sys?$top=4&ver=11.5.0",
  "selfLink" : "https://localhost/mgmt/tm/sys?$top=4&$skip=4&ver=11.5.0",
  "startIndex" : 5,
  "totalItems" : 36,
  "totalPages" : 9
}
```

About sub-collection expansion

iControl® REST supports the `expandSubcollections` query parameter. In `tmsh`, configuration components contain properties, child components, and associated, non-child components. For example, you can create an associated component independently from the component that contains it, such as a virtual server (the `ltm virtual` component in `tmsh`) that contains an LTM® pool, even though you create the LTM pool as a separate task.

If set to `true`, the `expandSubcollections` query parameter displays all child components but omits any associated non-child components the response.

Although the command creates a lengthy output block, the query parameter displays the properties of the sub-collection, in addition to the properties of the component. As with other query parameters, the `expandSubcollections` parameter does not support requests other than a GET request.

```
https://192.168.25.42/mgmt/tm/ltm/virtual/my-VS/?expandSubcollection=true
```

```
{
    "kind":"tm:ltm:virtual:virtualstate",
    "name":"my-VS",
    "fullPath":"my-VS",
    "generation":1,

"selfLink":"https://../tm/ltm/virtual/my-VS?expandSubcollections=true&ver=11.5.0",

    "autoLasthop":"default",
    "cmpEnabled":"yes",
    "connectionLimit":0,
    "destination":"/Common/10.2.1.189:0",
    "enabled":null,
    "gtmScore":0,
    "ipProtocol":"tcp",
    "mask":"255.255.255.255",
    "mirror":"disabled",
    "mobileAppTunnel":"disabled",
    "nat64":"disabled",
    "pool":"/Common/my-Pool",
    "rateLimit":"disabled",
    "rateLimitDstMask":0,
    "rateLimitMode":"object",
    "rateLimitSrcMask":0,
    "source":"0.0.0.0/0",
    "sourceAddressTranslation":{
        "type":"automap"
    },
    "sourcePort":"preserve",
    "synCookieStatus":"not-activated",
    "translateAddress":"enabled",
    "translatePort":"disabled",
    "vlansDisabled":null,
    "vsIndex":2,
    "policiesReference":{
        "link":"https://../tm/ltm/virtual/~Common~my-VS/policies?ver=11.5.0",
        "isSubcollection":true,
        "items":[
            {
                "kind":"tm:ltm:virtual:policies:policiesstate",
                "name":"asm_auto_l7_policy__my-VS",
                "partition":"Common",
                "fullPath":"/Common/asm_auto_l7_policy__my-VS",
                "generation":1,
```

```
"selfLink":"https://../~Common~my-VS/policies/~Common~asm_auto_l7_policy__my-VS?ver=11.5.0"

            }
        ]
    },
    "securityLogProfiles":[
        "\"/Common/Log illegal requests\""
    ],
    "fwRulesReference":{
        "link":"https://../tm/ltm/virtual/~Common~my-VS/fw-rules?ver=11.5.0",
        "isSubcollection":true
    },
    "profilesReference":{
        "link":"https://../tm/ltm/virtual/~Common~my-VS/profiles?ver=11.5.0",
        "isSubcollection":true,
        "items":[
            {
                "kind":"tm:ltm:virtual:profiles:profilesstate",
                "name":"http",
                "partition":"Common",
                "fullPath":"/Common/http",
                "generation":1,

"selfLink":"https://../tm/ltm/virtual/~Common~my-VS/profiles/~Common~http?ver=11.5.0",

                "context":"all"
            },
            {
                "kind":"tm:ltm:virtual:profiles:profilesstate",
                "name":"tcp",
                "partition":"Common",
                "fullPath":"/Common/tcp",
                "generation":1,

"selfLink":"https://../tm/ltm/virtual/~Common~my-VS/profiles/~Common~tcp?ver=11.5.0",

                "context":"all"
            },
            {
                "kind":"tm:ltm:virtual:profiles:profilesstate",
                "name":"websecurity",
                "partition":"Common",
                "fullPath":"/Common/websecurity",
                "generation":1,

"selfLink":"https://../tm/ltm/virtual/~Common~my-VS/profiles/~Common~websecurity?ver=11.5.0",

                "context":"all"
            }
        ]
    }
}
```

Expanding a sub-collection reference

The responses from iControl® REST can include references to sub collections. The expandSubcollections query parameter expands references to sub-collections.

View the details of a particular resource, including the details of its sub-collections, append the string expandSubcollections=true to the URI. Do not prepend a dollar sign ($) to this query parameter.

To see the differences, this example shows a GET request for a resource with sub-collection expansion. The response contains the `isSubcollection` property, set to true, to indicate a sub-collection. The output only contains a reference to the sub-collection.

```
https://192.168.42.25/mgmt/tm/ltm/pool/~Common~my-Pool
```

```
{ "allowNat" : "yes",
  "allowSnat" : "yes",
  "description" : "sdfds",
  "fullPath" : "/Common/my-Pool",
  "generation" : 1,
  "ignorePersistedWeight" : "disabled",
  "ipTosToClient" : "pass-through",
  "ipTosToServer" : "pass-through",
  "kind" : "tm:ltm:pool:poolstate",
  "linkQosToClient" : "pass-through",
  "linkQosToServer" : "pass-through",
  "loadBalancingMode" : "round-robin",
  "membersReference" : { "isSubcollection" : true,
      "link" : "https://../mgmt/tm/ltm/pool/~Common~my-Pool/members?ver=11.5.0"

    },
  "minActiveMembers" : 0,
  "minUpMembers" : 0,
  "minUpMembersAction" : "failover",
  "minUpMembersChecking" : "disabled",
  "name" : "my-Pool",
  "partition" : "Common",
  "queueDepthLimit" : 0,
  "queueOnConnectionLimit" : "disabled",
  "queueTimeLimit" : 0,
  "reselectTries" : 0,
  "selfLink" : "https://../mgmt/tm/ltm/pool/~Common~my-Pool?ver=11.5.0",
  "slowRampTime" : 10
}
```

To see the expanded sub-collection, this example uses the `expandSubcollections` query parameter. iControl® REST supports the custom `expandSubcollections` query parameter, which omits the dollar sign ($) from its name.

```
https://192.168.25.42/mgmt/tm/ltm/pool/~Common~my-Pool/?expandSubcollections=true
```

```
{ "allowNat" : "yes",
  "allowSnat" : "yes",
  "description" : "sdfds",
  "fullPath" : "/Common/my-Pool",
  "generation" : 1,
  "ignorePersistedWeight" : "disabled",
  "ipTosToClient" : "pass-through",
  "ipTosToServer" : "pass-through",
  "kind" : "tm:ltm:pool:poolstate",
  "linkQosToClient" : "pass-through",
  "linkQosToServer" : "pass-through",
  "loadBalancingMode" : "round-robin",
  "membersReference" : { "isSubcollection" : true,
      "items" : [ { "address" : "1.1.1.1",
            "connectionLimit" : 0,
            "dynamicRatio" : 1,
            "fullPath" : "/Common/block:0",
            "generation" : 1,
```

```
                "inheritProfile" : "enabled",
                "kind" : "tm:ltm:pool:members:membersstate",
                "logging" : "disabled",
                "monitor" : "default",
                "name" : "block:0",
                "partition" : "Common",
                "priorityGroup" : 0,
                "rateLimit" : "disabled",
                "ratio" : 1,
                "selfLink" :
"https://../tm/ltm/pool/~Common~my-Pool/members/~Common~block:0?ver=11.5.0",
                "session" : "user-enabled",
                "state" : "unchecked"
            } ],
        "link" : "https://../tm/ltm/pool/~Common~my-Pool/members?ver=11.5.0"
    },
   "minActiveMembers" : 0,
   "minUpMembers" : 0,
   "minUpMembersAction" : "failover",
   "minUpMembersChecking" : "disabled",
   "name" : "my-Pool",
   "partition" : "Common",
   "queueDepthLimit" : 0,
   "queueOnConnectionLimit" : "disabled",
   "queueTimeLimit" : 0,
   "reselectTries" : 0,
   "selfLink" :
"https://../tm/ltm/pool/~Common~my-Pool?expandSubcollections=true&ver=11.5.0",

   "slowRampTime" : 10
}
```

Returning resources from an administrative partition

To access an administrative partition, use the `$filter` query parameter in a GET request to specify a resource in a partition.

1. Access a partition other than `Common`, using the `$filter` query option at the end of the URI.

2. Encode the URI by creating the following string: `?$filter=partition%20eq%20fw_objs`

To use a filter parameter, this example shows a GET request that uses a filter setting to limit the query to a specific partition. The response from the request appears in the second block.

```
GET https://192.168.25.42/mgmt/tm/ltm/pool/?$filter=partition eq fw_objs
```

```
{
    "kind":"tm:ltm:pool:poolcollectionstate",

"selfLink":"https://../mgmt/tm/ltm/pool?$filter=partition%20eq%20fw_objs&ver=11.5.0",

    "items":[
        {
            "kind":"tm:ltm:pool:poolstate",
            "name":"tcb-pool2",
            "partition":"fw_objs",
            "fullPath":"/fw_objs/tcb-pool2",
            "generation":9587,
```

```
          "selfLink":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2?ver=11.5.0",

          "allowNat":"yes",
          "allowSnat":"yes",
          "description":"This pool exists in the fw_objs partition.",
          "ignorePersistedWeight":"disabled",
          "ipTosToClient":"pass-through",
          "ipTosToServer":"pass-through",
          "linkQosToClient":"pass-through",
          "linkQosToServer":"pass-through",
          "loadBalancingMode":"round-robin",
          "minActiveMembers":0,
          "minUpMembers":0,
          "minUpMembersAction":"failover",
          "minUpMembersChecking":"disabled",
          "queueDepthLimit":0,
          "queueOnConnectionLimit":"disabled",
          "queueTimeLimit":0,
          "reselectTries":0,
          "slowRampTime":10,
          "membersReference":{

"link":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2/members?ver=11.5.0",
             "isSubcollection":true
          }
        }
    ]
}
```

Obtaining statistical output

The response to a GET request contains the JSON representations that are equivalent to the output of the tmsh list command. iControl® REST includes a suffix for statistical information that produces statistical output equivalent to the tmsh show command.

1. Obtain statistical results for a resource by specifyi the resource of interest in the URI.

2. Append the endpoint stats to the URI of the resource to obtain statistical output.

To obtain statistics for a resource, this example queries the /Common/my-Pool object for current statistics. The response that contains the statistical output appears in the second block.

```
GET https://192.168.25.42/mgmt/tm/ltm/pool/~Common~my-Pool/stats
```

```
{
   "kind":"tm:ltm:pool:poolstats",
   "generation":1,
   "selfLink":"https://../mgmt/tm/ltm/pool/~Common~my-Pool/stats?ver=11.5.0",

   "entries":{
      "activeMemberCnt":{
         "value":0
      },
      "connqAll.ageEdm":{
         "value":0
      },
      "connqAll.ageEma":{
```

```
        "value":0
    },
    "connqAll.ageHead":{
        "value":0
    },
    "connqAll.ageMax":{
        "value":0
    },
    "connqAll.depth":{
        "value":0
    },
    "connqAll.serviced":{
        "value":0
    },
    "connq.ageEdm":{
        "value":0
    },
    "connq.ageEma":{
        "value":0
    },
    "connq.ageHead":{
        "value":0
    },
    "connq.ageMax":{
        "value":0
    },
    "connq.depth":{
        "value":0
    },
    "connq.serviced":{
        "value":0
    },
    "curSessions":{
        "value":0
    },
    "minActiveMembers":{
        "value":0
    },
    "monitorRule":{
        "description":"none"
    },
    "tmName":{
        "description":"/Common/my-Pool"
    },
    "serverside.bitsIn":{
        "value":0
    },
    "serverside.bitsOut":{
        "value":0
    },
    "serverside.curConns":{
        "value":0
    },
    "serverside.maxConns":{
        "value":0
    },
    "serverside.pktsIn":{
        "value":0
    },
    "serverside.pktsOut":{
        "value":0
    },
    "serverside.totConns":{
        "value":0
    },
    "status.availabilityState":{
        "description":"unknown"
    },
```

```
      "status.enabledState":{
         "description":"enabled"
      },
      "status.statusReason":{
         "description":"The children pool member(s) either don't have service
 checking enabled, or service check results are not available yet"
      },
      "totRequests":{
         "value":0
      }
   }
}
```

For comparison, this is the response from the tmsh command to display statistical output.

```
root@(BIG-IP1)(...)(tmos)# show ltm pool my-Pool

-----------------------------------------------------------------------
Ltm::Pool: my-Pool
-----------------------------------------------------------------------
Status
  Availability : unknown
  State        : enabled
  Reason       : The children pool member(s) either don't have service checking
 enabled, or service check results are not available yet
  Monitor      : none
  Minimum Active Members : 0
  Current Active Members : 0
          Total Requests : 0
        Current Sessions : 0

Traffic                                         ServerSide
  Bits In                                             0
  Bits Out                                            0
  Packets In                                          0
  Packets Out                                         0
  Current Connections                                 0
  Maximum Connections                                 0
  Total Connections                                   0

Connection Queue                                Pool  Pool and members
  Number of connections queued now               0                   0
  Number of connections serviced                 0                   0
  Queue head entry age (ms)                      0                   0
  Maximum queue entry age ever (ms)              0                   0
  Maximum queue entry age recently (ms)          0                   0
  Average queue entry age (ms)                   0                   0

root@(BIG-IP1)(...)(tmos)#
```

Chapter

3

POST and PUT requests

- *About JSON format for POST and PUT*
- *Adding and modifying resources*

About JSON format for POST and PUT

Unlike a GET request, a POST or PUT request includes a JSON body. When you create or modify a resource, you use the same JavaScript Object Notation (JSON) format as shown in a GET request to define the configuration of an object. Use POST to create a new configuration object from a JSON body, and use PUT or PATCH to edit an existing configuration object with a JSON body.

The format of the JSON body consists of objects that follow the model for an object, as shown:

```
{ "partition":"Common" }
```

Both the name and value appear in double quotes, and a colon separates the name and the value in the pair. For objects that contain multiple name pairs, a comma (,) separates additional name/value pairs. A JSON value must be an object, array, number, string, or one of three literal names: `false`, `null`, or `true`. The other structure is a JSON array, or collection, which is an ordered list of values, as shown:

```
[ { "components":8, "isSubcomponent":"true" } ]
```

In JSON format, square brackets enclose the objects in an array. The objects in the array follow the JSON standard for name/value pairs. Collectively, the name/value pairs are the properties of a BIG-IP® system configuration. For iControl REST, the name/value pairs can be thought of as property name and property value.

In a REST call, declare the format of the object to post . For iControl REST, specify the format `application/json`. In a `curl` command, for example, specify the HTTP header `-H "Content-Type: application/json"` to declare JSON format:

```
curl -k -u username:password -H "Content-Type: application/json"
             -X http-method uri
```

Within the JSON body, define the name of the configuration object. Then include the property names and values for the object, using the same names and properties that appear in the response to a GET request for a similar object. Any properties that you omit revert to the existing values, for a a PUT request, or their default values, for a POST request. If you use a tool like curl, you can specify the JSON body in the command line. Several examples in this guide demonstrate the inclusion of a JSON body from the command line.

Adding and modifying resources

The iControl® REST API enables you to add resources to a BIG-IP® system. To add a resource, use the POST method on an iControl® REST collection and specify the resource to create as a JSON body. When you create a resource, iControl® REST sets all unspecified properties to their default values.

1. To add a new configuration object to a collection by specifying the path to the collection in the URI.
2. Specify the name of the object to create, as a property name/value pair.
3. Add one or more resources to the parent sub-collection, as needed.

To demonstrate the use of POST, this example creates a new pool in the `pool` collection. The response to the request is shown in the second block. Note that iControl® REST sets all of the default properties for the configuration object.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X POST -d \
    '{"name":"tcb-pool","members":[ \
    {"name":"192.168.25.32:80","description":"first member"} ]' \
     https://192.168.25.42/mgmt/tm/ltm/pool
```

The JSON body contains the name/value pairs for the `name`, `partition`, and pool `members`.

```
{"name":"tcb-pool2", "partition":"Common",
"members":[{"name":"192.168.25.32:80", "description":"Web server"}]}
```

The response to the request shows the properties for the tcb-pool2 resource, including a link to the sub-collection.

```
{
    "kind":"tm:ltm:pool:poolstate",
    "name":"tcb-pool2",
    "partition":"Common",
    "fullPath":"/Common/tcb-pool2",
    "generation":57,
    "selfLink":"https://localhost/mgmt/tm/ltm/pool/~Common~tcb-pool2?ver=11.5.0",

    "allowNat":"yes",
    "allowSnat":"yes",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{

"link":"https://localhost/mgmt/tm/ltm/pool/~Common~tcb-pool2/members?ver=11.5.0",

        "isSubcollection":true
    }
}
```

Modifying a configuration object

The PUT method allows modifications to properties of a configuration object without affecting any other properties.

POST and PUT requests

1. To modify a configuration object by specifying the configuration object itself in the URI. Do not specify a collection.
2. Specify the properties to modify as name/value pairs in the JSON body.

To modify a resource, make a PUT request with a JSON body. The response that appears in the second block illustrates the changes to the properties.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X PUT -d \
    '{"name":"tcb-pool","description":"backup web \
    servers","ignorePersistedWeight": "enabled"}' \
    https://192.168.25.42/mgmt/tm/ltm/pool/tcb-pool
```

```
{
    "kind":"tm:ltm:pool:poolstate",
    "name":"tcb-pool",
    "fullPath":"tcb-pool",
    "generation":2085,
    "selfLink":"https://../mgmt/tm/ltm/pool/tcb-pool?ver=11.5.0",
    "allowNat":"yes",
    "allowSnat":"yes",
    "description":"backup member",
    "ignorePersistedWeight":"enabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{
      "link":"https://../mgmt/tm/ltm/pool/~Common~tcb-pool/members?ver=11.5.0",

      "isSubcollection":true
    }
}
```

To modify a resource, make a PATCH request with a JSON body. The response that appears in the second block illustrates the changes to the properties.

```
PATCH https://192.168.25.42/mgmt/tm/pool/~Common~tcb-pool2
```

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X PUT -d \
    '{"name":"tcb-pool2","member": [ {:name":"192.168.25.32:80",
"description":"Tertiary web server"}] }' \
    https://192.168.25.42/mgmt/tm/ltm/pool/tcb-pool2
```

```
{
    "kind":"tm:ltm:pool:poolstate",
```

```
    "name":"tcb-pool2",
    "partition":"Common",
    "fullPath":"/Common/tcb-pool2",
    "generation":59,
    "selfLink":"https://../mgmt/tm/ltm/pool/~Common~tcb-pool2?ver=11.5.0",
    "allowNat":"yes",
    "allowSnat":"yes",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{
      "link":"https://../mgmt/tm/ltm/pool/~Common~tcb-pool2/members?ver=11.5.0",

       "isSubcollection":true
    }
}
```

Modifying a collection

Configuration objects can contain collections, denoted by square brackets in JSON format. For example, an LTM® pool has a collection of member objects, and a rule-list (in the security firewall module) has a collection of rule objects. The actions of a PUT method depend greatly on the Request-URI. When you use the PUT method with a collection, only the properties that you specify in the JSON body will exist in the modified collection. In other words, the PUT method overwrites any existing properties of the collection. Therefore, if you use the PUT method to modify one member object of an LTM pool that originally had three members, the request reduces the collection to just the single member.

1. To modify a collection, specify the collection in the URI.
2. Specify the properties of the collection, as name/value pairs in the JSON body.

To make modifications to the sub-collection, make a PUT request to add a primary member and a backup member to a sub-collection. The semantics of a PUT request replace the member collection if you do not specify any existing members in the JSON body. If the member collection already exists, this request overwrites the properties of the sub-collection.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X PUT -d \
    '{"name":"tcb-pool","members":[ \
    {"name":"192.168.25.32:80","description": "primary member"}, \
    {"name":"192.168.25.33:80","description":"backup member}]}' \
    https://192.168.25.42/mgmt/tm/ltm/pool/tcb-pool
```

```
{
    "kind":"tm:ltm:pool:poolstate",
```

```
    "name":"tcb-pool",
    "fullPath":"tcb-pool",
    "generation":258,
    "selfLink":"https://localhost/mgmt/tm/ltm/pool/tcb-pool?ver=11.5.0",
    "allowNat":"yes",
    "allowSnat":"yes",
    "description":"# Paging server #",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{
      "link":"https://../mgmt/tm/ltm/pool/~Common~tcb-pool/members?ver=11.5.0",

       "isSubcollection":true
    }
}
```

The changes to the members sub-collection appear as a link in the response in iControl® REST. To view the changes, either make a GET request and specify the member collection in the URI.

```
GET https://192.168.25.42/mgmt/tm/ltm/pool/tcb-pool/members
```

```
{
    "kind":"tm:ltm:pool:members:memberscollectionstate",
    "selfLink":"https://localhost/mgmt/tm/ltm/pool/tcb-pool/members?ver=11.5.0",

    "items":[
        {
            "kind":"tm:ltm:pool:members:membersstate",
            "name":"192.168.25.32:80",
            "partition":"Common",
            "fullPath":"/Common/192.168.25.32:80",
            "generation":2133,

"selfLink":"https://../pool/tcb-pool/members/~Common~192.168.25.32:80?ver=11.5.0",

            "address":"192.168.25.32",
            "connectionLimit":0,
            "description":"primary member",
            "dynamicRatio":1,
            "inheritProfile":"enabled",
            "logging":"disabled",
            "monitor":"default",
            "priorityGroup":0,
            "rateLimit":"disabled",
            "ratio":1,
            "session":"user-enabled",
            "state":"unchecked"
        },
        {
            "kind":"tm:ltm:pool:members:membersstate",
            "name":"192.168.25.33:80",
```

```
            "partition":"Common",
            "fullPath":"/Common/192.168.25.33:80",
            "generation":2133,

 "selfLink":"https://../pool/tcb-pool/members/~Common~192.168.25.33:80?ver=11.5.0",

            "address":"192.168.25.33",
            "connectionLimit":0,
            "description":"backup member",
            "dynamicRatio":1,
            "inheritProfile":"enabled",
            "logging":"disabled",
            "monitor":"default",
            "priorityGroup":0,
            "rateLimit":"disabled",
            "ratio":1,
            "session":"user-enabled",
            "state":"unchecked"
        }
    ]
}
```

Modifying a single object in a collection

A subcollection permits direct access to individual items, eliminating the need to specify all members of the collection when making a single change.

Specify the individual item, by name, to modify an item in a collection.

This example shows the use of a request with the PUT method to modify a single object in a collection. The URI specifies the individual member object to modify, and the modification consists of a single change to a property. The resulting change is shown in the block.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X PUT -d '{"description": "secondary, not backup,
member"}'\
    https://192.168.25.42/mgmt/tm/ltm/pool/tcb-pool/members/192.168.25.33:80
```

```
{
   "kind":"tm:ltm:pool:members:memberscollectionstate",
   "selfLink":"https://localhost/mgmt/tm/ltm/pool/tcb-pool/members?ver=11.5.0",

   "items":[
       {
           "kind":"tm:ltm:pool:members:membersstate",
           "name":"192.168.25.32:80",
           "partition":"Common",
           "fullPath":"/Common/192.168.25.32:80",
           "generation":2133,

 "selfLink":"https://../pool/tcb-pool/members/~Common~192.168.25.32:80?ver=11.5.0",

           "address":"192.168.25.32",
           "connectionLimit":0,
           "description":"primary member",
           "dynamicRatio":1,
           "inheritProfile":"enabled",
           "logging":"disabled",
```

```
           "monitor":"default",
           "priorityGroup":0,
           "rateLimit":"disabled",
           "ratio":1,
           "session":"user-enabled",
           "state":"unchecked"
       },
       {
           "kind":"tm:ltm:pool:members:membersstate",
           "name":"192.168.25.33:80",
           "partition":"Common",
           "fullPath":"/Common/192.168.25.33:80",
           "generation":2156,

 "selfLink":"https://../pool/tcb-pool/members/~Common~192.168.25.33:80?ver=11.5.0",

           "address":"192.168.25.33",
           "connectionLimit":0,
           "description":"secondary, not backup, member",
           "dynamicRatio":1,
           "inheritProfile":"enabled",
           "logging":"disabled",
           "monitor":"default",
           "priorityGroup":0,
           "rateLimit":"disabled",
           "ratio":1,
           "session":"user-enabled",
           "state":"unchecked"
       }
   ]
}
```

About read only properties

If you specify a read only property with a PUT or POST method, iControl® REST accepts the request and generates an error response. If you specify other properties in addition to the read only property, a valid PUT or POST request will not generate an error, despite the inclusion of the read only property, .

For example, the following curl command specifies a read only property in an existing cm device object: timeZone. The response from iControl® REST indicates a missing property name. In this situation, iControl® REST ignores the read only property and generates the error message shown in the second block.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X PUT -d \
    '{"time-zone":"EDT"}' \
    https://192.168.25.42/mgmt/tm/cm/device/bigip1
```

```
{
    "code":400,
    "message":"one or more properties must be specified",
    "errorStack":[

    ]
}
```

Adding or modifying in a specific partition

To add or modify a resource in an administrative partition, add the partition property to the JSON body to modify configuration objects. Use the query option on the command line, or include a `partition` property in the JSON body. Keep in mind that the $filter query parameter applies to GET requests only.

To modify a configuration object with a PUT method, identify the object's partition in the `partition` property.

This example uses the POST method to create a resource in a partition other than the Common partition. Specify the name of the resource, and the partition in which to create it, in the JSON body. The response to the request is shown in the third block.

```
POST https://192.168.25.42/mgmt/tm/ltm/pool
```

```
{ "name":"tcb-pool2", "partition":"~fw_objs" }
```

```
{
    "kind":"tm:ltm:pool:poolstate",
    "name":"tcb-pool2",
    "partition":"fw_objs",
    "fullPath":"/fw_objs/tcb-pool2",
    "generation":7810,

"selfLink":"https://localhost/mgmt/tm/ltm/pool/~fw_objs~tcb-pool2?ver=11.5.0",

    "allowNat":"yes",
    "allowSnat":"yes",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{

"link":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2/members?ver=11.5.0",
        "isSubcollection":true
    }
}
```

Following the creation of a new configuration object, this example modifies the member collection by using a PUT request. The URI includes the full path to the resource to modify. Specify the partition property, as

well as any properties you wish to modify. The partition property in the JSON body matches the folder name. The response to the request is shown in the third block.

```
PUT https://192.168.25.42/mgmt/tm/ltm/pool/~fw_objs~tcb-pool2
```

```
{ "name":"tcb-pool2", "partition":"/fw_objs",
"members": [ {"name":"192.168.25.32", "description":"Marketing server"} ] }
```

```
{
    "kind":"tm:ltm:pool:poolstate",
    "name":"tcb-pool2",
    "partition":"fw_objs",
    "fullPath":"/fw_objs/tcb-pool2",
    "generation":7914,

"selfLink":"https://localhost/mgmt/tm/ltm/pool/~fw_objs~tcb-pool2?ver=11.5.0",

    "allowNat":"yes",
    "allowSnat":"yes",
    "description":"This pool exists in the fw_objs partition.",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{

"link":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2/members?ver=11.5.0",
        "isSubcollection":true
    }
}
```

About relative partitions and folder names

If you use a relative folder path within a partition body, iControl®REST interprets the folder name relative to the parent partition. Set the parent partition with the by specifying the $filter=partition eq *folder-name* query parameter in the URI, or the partition property in the JSON body, depending on the type of request. The $filterquery parameter applies to GET requests, whereas the partition property applies to POST or PUT requests. For example, if the $filter=partition query option is set to /eu and the JSON body includes a reference to the france folder, iControl® REST interprets the folder path as /eu/france. To avoid ambiguity with partition and folder names, use absolute paths for all folders in JSON body, such as /eu/france.

The $filter query parameter differs from the OData query parameter in that it only supports filtering by partition names in iControl® REST.

Deleting a configuration object

The HTTP DELETE method removes single resources from a BIG-IP® system configuration. The semantics of the DELETE method support the deletion of a resource, but iControl® REST does not support the use of DELETE to remove collections.

To determine the resource to delete, use the GET method to confirm that the resource you want to delete currently exists.

This example uses the GET method to get the configuration from a BIG-IP® system. In this case, the resource exists in the /fw_objs folder. The response to the request is shown in the second block.

```
GET https://192.168.25.42/mgmt/tm/ltm/pool/~fw_objs~tcb-pool2
```

```
{
    "kind":"tm:ltm:pool:poolstate",
    "name":"tcb-pool2",
    "partition":"fw_objs",
    "fullPath":"/fw_objs/tcb-pool2",
    "generation":1,
    "selfLink":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2?ver=11.5.0",
    "allowNat":"yes",
    "allowSnat":"yes",
    "description":"This pool exists in the fw_objs partition.",
    "ignorePersistedWeight":"disabled",
    "ipTosToClient":"pass-through",
    "ipTosToServer":"pass-through",
    "linkQosToClient":"pass-through",
    "linkQosToServer":"pass-through",
    "loadBalancingMode":"round-robin",
    "minActiveMembers":0,
    "minUpMembers":0,
    "minUpMembersAction":"failover",
    "minUpMembersChecking":"disabled",
    "queueDepthLimit":0,
    "queueOnConnectionLimit":"disabled",
    "queueTimeLimit":0,
    "reselectTries":0,
    "slowRampTime":10,
    "membersReference":{
 "link":"https://../mgmt/tm/ltm/pool/~fw_objs~tcb-pool2/members?ver=11.5.0",
        "isSubcollection":true
    }
}
```

This example uses the DELETE method to delete the resource. iControl® REST generates a response code but no other output for a successful delete request.

```
DELETE https://192.168.25.42/mmt/tm/ltm/pool/~fw_objs~tcb-pool2
```

Chapter

4

Partitions

- *About administrative partitions*

About administrative partitions

Many types of BIG-IP® system objects, such as profiles and pools, reside in administrative partitions. Partitions are containers with administrative boundaries that you control with access permissions. Through restricted access to adminstrative partitions, you impose greater control over the configuration objects, and reduce the likelihood of inadvertent changes to the system configuration.

The Common partition contains all default profiles, preconfigured monitors, default authentication iRules, the root and admin user accounts, and route domain 0, which is the default route domain. The Common partition is created by the BIG-IP® installation process. If there are no other administrative partitions on a system, all objects that you create will exist in the Common partition. All administrators can access the Common partition.

Administrators that have the Administrator or Resource Administrator role associated with their user account can create partitions. When you create other partitions, you can associate a user account to that partition and grant permissions to administer that partition. In most circumstances, you either grant a user access to a single partition or universal access to all partitions, A user with access to a single partition can only create objects in that partition. If you grant a user universal access to all partitions, the user must select the partition in which to create an object by specifying the sys/folder namespace and the folder name in the request URI.

Every partition has a corresponding folder in the sys/folder namespace, including the Common partition which has an associated /Common folder. You can specify a namespace in an iControl® REST URI when you create or delete a partition. For more information about configuring administrative partitions and role based access control (RBAC), see the topic *Configuring Administrative Partitions*.

Important: *You cannot remove the Common partition, regardless of your level of administrative access.*

Adding an administrative partition

To add an administrative partition, you must have the Administrator or Resource Administrator role associated with your user account. Every administrative partition has an associated folder, and the sys/folder namespace indicates a partition to iControl® REST.

Important: *You must assign user permissions to the partition through a separate request to iControl® REST.*

Use the POST operation to create to a new partition in the sys/folder module. Specify the folder name, including the leading slash (/) character, as the value of the name property in the JSON body.

This iControl® REST request adds the fw_objs partition to a BIG-IP® system configuration. The second code block contains the response to the request.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X POST -d '{"name":"/fw_objs"}' \
```

```
    https://192.168.25.42/mgmt/tm/sys/folder \
    |python -m json.tool
```

```
{
   "kind":"tm:sys:folder:folderstate",
   "name":"fw_objs",
   "subPath":"/",
   "fullPath":"/fw_objs",
   "generation":2014,
   "selfLink":"https://localhost/mgmt/tm/sys/folder/~fw_objs?ver=11.5.0",
   "deviceGroup":"none",
   "hidden":"false",
   "inheritedDevicegroup":"true",
   "inheritedTrafficGroup":"true",
   "noRefCheck":"false",
   "trafficGroup":"/Common/traffic-group-1"
}
```

Deleting an administrative partition

An administrative partition, other than Common, can be deleted with a DELETE request. In the URI, specify the folder name of the partition to delete, and submit the request without a JSON body. Because a folder name includes a forward slash, the folder name must be specified with a tilde character.

Important: You can only delete a partition if it is empty. Remove all objects in the partition before you attempt to delete the partition.

To delete a partitionspecify the DELETE method and the folder namespace /mgmt/tm/sys/folder/ in the URI. Replace each forward slash (/) in the folder name with a tilde character (~).

In this example, the iControl® REST request deletes the /fw_objs partition from the system configuration. The response includes a response code to indicate success or failure, but the response does not produce a JSON body unless there is an error in the request.

```
curl -k -u admin:admin -H "Content-Type: \
    application/json" -X DELETE \
    https://192.168.25.42/mgmt/tm/sys/folder/~fw_objs \
    |python -m json.tool
```

Chapter

5

Transactions

- *About the transaction model*
- *Creating a transaction*

About the transaction model

iControl® REST supports batch mode transactions as a sequence of web service requests. The basic iControl REST methods to create, delete, commit, or query a resource also apply to transactions. However, instead of individual method requests, you combine multiple requests into a transaction, and perform the sequence of requests atomically. The transaction completes successfully if all of the commands in the transaction complete successfully. Otherwise, the transaction fails and the commands in the transaction are rolled back, or not committed.

A transaction is a root level collection, similar to the `sys` or `ltm`. To manage and create transactions, use the iControl REST transaction resource `https://localhost/mgmt/tm/transaction`. When you create a transaction, the transaction resource associates three properties with that transaction:

- A `transId` property that identifies a transaction for the life of the transaction.
- A `state` property that indicates the state of the transaction. Values for this property are: STARTED, UPDATING, VALIDATING, COMPLETED, and FAILED.
- A `timeoutSeconds` property that specifies the time period during which to add commands to the transaction. The default value is 30 seconds.

The initial step to create a transaction requires that you specify the iControl REST transaction collection with a POST method. This method results in the creation of a transaction resource with the properties mentioned previously. Of the three properties, the timeout property represents a constraint on the amount of time you have to add commands to the transaction. If you do not complete the modifications of a command sequence before the timeout value occurs, the transaction resource is deleted. After you create a transaction, use the POST, PATCH, or PUT methods and include the transaction ID in the HTTP header of the request to add commands to the transaction. The final step to commit a transaction involves a PUT or PATCH method to modify the transaction state property to `VALIDATING` to commit the transaction. If the transaction runs successfully, all pending REST operations are cleared from the server and the transaction is deleted. The transaction resource limits the amount of time that a transaction exists. If the command sequence is delayed by a command that takes an inordinate amount of time to finish, the transaction resource may be deleted.

Method	Description
POST	To create a transaction, specify the URI `https://localhost/mgmt/tm/transaction` with an empty JSON message body. The response contains the following properties: • transId • state • timeoutSeconds
DELETE	To delete a transaction, specify the URI `https://localhost/mgmt/tm/transaction/transId` where `transId` is the identifier for the transaction. iControl REST deletes all operations associated with this transaction.
PATCH, PUT	To commit a transaction, specify the URI `https://localhost/mgmt/tm/transaction/transId`, where `transId` is the identifier for the transaction. In the JSON message body, specify a key/value pair `"state":"VALIDATING"`
GET	To query the transaction collection, specify the URI `https://localhost/mgmt/tm/transaction`.

Method	Description
GET	To query a single transaction, specify the URI `https://localhost/mgmt/tm/transaction/transId`, where `transId` is the identifier for the transaction.

After you create a transaction, you can add commands to that transaction. In order to do this, specify the `X-F5-REST-Coordination-Id` coordination header and the `transId` to identify the transaction to receive the command. When you make a POST or PUT request with the HTTP header and the transaction identifier, any commands are queued to the transaction identified by `transId`. The properties of the request are defined in a JSON body, as with any iControl® REST request. Each request made with the coordination header adds an additional request to the transaction. A transaction resource stores and runs the commands in the order that you add them. You can use the DELETE, PATCH, POST, and PUT methods in a transaction just as you do with any iControl® REST request. GET requests with a coordination header are ignored and never included in a transaction.

The commands of a transaction are identified by two values, a command identifier and an evaluation order identifier. The command identifier and evaluation order identifiers are known as `commandId` and `evalOrder`. iControl REST assigns the `commandId` and `evalOrder` as you add the commands to a transaction. The command identifier identifies the commands in the order that you added them to the transaction. The evaluation order identifier specifies the order of the commands as the transaction submits them. Unless you modify the order of the commands, they execute in the order that you submitted them. You can modify the evaluation order that the transaction submits the commands, but you must do so within the time period allocated for the transaction.

Method	Description
GET	To obtain the details of a single command, specify the URI `https://localhost/mgmt/tm/transaction/transId/commands/commandId`, where the `transId` is the identifier for the transaction, and `commandId` is the identifier for the command.
GET	To obtain a list of commands in a transaction, specify the URI `https://localhost/mgmt/tm/transaction/transId/commands`, where `transId` is the identifier for the transaction.
DELETE	To remove a command from a transaction, specify the URI `https://localhost/mgmt/tm/transaction/transId/commands/commandId`, where `transId` is the identifier for the transaction, and `commandId` is the identifier for the command. The remaining commands in the transaction will be renumbered.
PATCH	To change the evaluation order, specify the URI `https://localhost/mgmt/tm/transaction/transId/commands/commandId`, where `transId` is the identifier for the transaction, and `commandId` is the identifier for the command. In the JSON message body, specify a key/value pair `"evalOrder":"y"`. This action moves the command.
PATCH	To rearrange commands, use the PATCH verb and specify the `transId`, `commandId`, and `evalOrder`.

Creating a transaction

Transactions allow you to run a sequence of commands atomically. By using a specific endpoint, you can create a transaction, then add commands to the transaction, and commit the transaction.

Use the POST method to create a transaction, and include the transaction ID to add commands to the transaction.

To create a transaction, use the POST method. The response includes the transaction identifier that must be present in the commands that you add to the transaction.

```
POST https://192.168.25.42/mgmt/tm/transaction
```

```
{
    "transId":1389812351,
    "state":"STARTED",
    "timeoutSeconds":30,
    "kind":"tm:transactionstate",
    "selfLink":"https://localhost/mgmt/tm/transaction/1389812351?ver=11.5.0"
}
```

To add a command to the transaction, use PATCH, POST, or PUT methods and specify the `X-F5-REST-Coordination-ID` header in the request.

```
POST https://192.168.25.42/mgmt/tm/transaction
```

```
{
    "name":"tcb-xact-pool",
    "members": [ {"name":"192.168.25.32:80","description":"First pool for
transactions"} ]
}
```

The response indicates that the command was added to the transaction.

```
{
    "transId":1389812351,
    "state":"STARTED",
    "timeoutSeconds":30,
    "kind":"tm:transactionstate",
    "selfLink":"https://localhost/mgmt/tm/transaction/1389813931?ver=11.5.0"
}
```

To commit the transaction, use the PATCH or PUT methods, and append the transaction ID to the URI. In the JSON body, specify the name/value pair shown in the code block.

```
PUT https://192.168.25.42/mgmt/tm/transaction/1389812351
```

```
{ "state":"VALIDATING" }
```

Chapter

6

Commands

- *About other tmsh global commands*

About other tmsh global commands

Not all *Traffic Management Shell (tmsh) Reference* commands map directly to HTTP methods. For a `list` or `show` request of a resource, a GET request maps well to the requested operation, but the reference includes global commands that do not directly correspond to an HTTP method. iControl® REST implements the following set of `tmsh` commands:

- `cp`
- `generate`
- `install`
- `load`
- `mv`
- `publish`
- `reboot`
- `restart`
- `reset_stats`
- `run`
- `save`
- `send-mail`
- `start`
- `stop`

iControl REST supports these `tmsh` commands by mapping a command, as well as options, to JSON format.

The iControl REST format for `tmsh` commands follows this general approach:

- Use the POST method.
- Specify a namespace for the `tmsh` command in the URI.
- Specify the command and options as the values of the properties in the JSON body.

To run the command, use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys/application/template`, along with the JSON body for the command. In each example, a relative URI is used in the request body.

Using the cp command

Utility commands do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys/application/template`, along with a JSON body that specifies the name of the utility command.

To copy using the `cp` command, make an iControl® REST request with the POST method and specify the properties in a JSON body.

To copy a file using the `cp` command, make a POST request. In the JSON body, specify the command, file name, and target file name.

```
POST /mgmt/tm/sys/application/template
```

```
{
    "command":"cp",
    "name":"tempt1",
    "target":"tempt2",
}
```

Using the generate command

Global commands like generate do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/ltm/rule`, along with a JSON body that specifies the name of the command.

To generate signed scripts using the `generate` command, make an iControl® REST request with the POST method and specify the properties in a JSON body.

To generate a signed script using the generate command, make a POST request. In the JSON body, specify the command, script name, options, and a signing key. The signing key property name uses a hyphenated name instead of the came case naming convention of iControl® REST.

```
POST /mgmt/tm/ltm/rule
```

```
{
    "command":"generate",
    "name":"rule1",
    "options":[
        {
            "signature":true
        }
    ],
    "signing-key":"key1"
}
```

Using the install command

Global commands like install do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys/software`, along with a JSON body that specifies the name of the command.

Install and update components using the `install` command by making an iControl® REST request with the POST method and a JSON body.

To install and update components using the install command, make a POST request. In the JSON body, specify the command, image, and volume.

```
POST /mgmt/tm/sys/software
```

```
{
    "command":"install",
    "image":"BIGIP-11.5.0.930.400.iso",
    "volume":"HD1.3"
}
```

To perform the same task and take advantage of the options for the install command, follow the previous steps and specify the create-volume and reboot options in the JSON body. The create volume property name uses a hyphenated name instead of the came case naming convention of iControl® REST.

```
POST /mgmt/tm/sys/software
```

```
{
    "command":"install",
    "options":[
        {
            "create-volume":true
        },
        {
            "reboot":true
        }
    ],
    "name":"BIGIP-11.4.0.737.400.42.iso",
    "volume":"HD1.1"
}
```

Using the load command

Global commands like load do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as https://192.168.25.42/mgmt/tm/sys/config, along with a JSON body that specifies the name of the command.

Load BIG-IP® system configuration using the load command by making an iControl® REST request with the POST method and a JSON body.

To replace the running configuration using the load command, make a POST request. In the JSON body, specify the command.

```
POST /mgmt/tm/sys/config
```

```
{
    "command":"load"
}
```

Using the mv command

Global commands like `mv` do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/cm/device`, along with a JSON body that specifies the name of the command.

To copy using the `mv` command, make an iControl® REST request with the POST method and specify the properties in a JSON body.

To move or rename an object using the `mv` command, make a POST request. In the JSON body, specify the command, name, and target:

```
POST /mgmt/tm/cm/device
```

```
{
    "command":"mv",
    "name":"bigip1",
    "target":"selfdevice2",
}
```

Using the publish command

Global commands, such as publish, do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/asm/policy`, along with a JSON body that specifies the name of the command.

Publish changes in a policy by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

In the JSON body, specify the command, name of the policy, and the application service. The application service property name uses a hyphenated name instead of the camel case naming convention of iControl REST.

```
POST /mgmt/tm/asm/policy
```

```
{
    "command":"publish",
    "name":"testpolicy",
    "app-service":"service",
}
```

Using the reboot command

Global commands like reboot do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys`, along with a JSON body that specifies the name of the command.

Reboot a system, or boot a system into a different volume by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To reboot a system using the reboot command, make a POST request. In the JSON body, specify the command.

```
POST /mgmt/tm/sys
```

```
{
    "command":"reboot"
}
```

Using the restart command

Global commands like restart do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys/service`, along with a JSON body that specifies the name of the command.

Restart a service by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To restart a service using the restart command, make a POST request. In the JSON body, specify the command and the name of the service to restart.

```
POST /mgmt/tm/sys/service
```

```
{
    "command":"restart",
    "name":"icrd"
}
```

Using the reset-stats command

Global commands like reset-stats do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/ltm/virtual`, along with a JSON body that specifies the name of the command.

Reset statistics for a component by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To reset statistics for a component using the reset-stats command, make a POST request. In the JSON body, specify the command and the name of the component.

```
POST /mgmt/tm/ltm/virtual
```

```
{
    "command":"reset_stats",
    "name":"http_vs1"
}
```

Using the run command

Global commands like run do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/util/ping`, along with a JSON body that specifies the name of the command.

Run a program by making an iControl® REST request with the POST method and specifying the properties in a JSON body. .

To run a command using the run command, make a POST request. In the JSON body, specify the command and the options for the command.

```
POST /mgmt/tm/util/ping
```

```
{
    "command":"run",
    "utilCmdArgs":"1.1.1.1 -c 1 -i 10"
}
```

Using the save command

Global commands like save do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as `https://192.168.25.42/mgmt/tm/sys/config`, along with a JSON body that specifies the name of the command.

Save the running configuration of a BIG-IP® system by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To save the running configuration using the save command, make a POST request. In the JSON body, specify the command.

```
POST /mgmt/tm/sys/config
```

```
{
    "command":"save"
}
```

To use the options available for the save command, specify the command and the options in a JSON body.

```
{
    "command":"save",
    "options":[
        {
            "file":"configfile.scf"
        }
    ]
}
```

Using the send-mail command

Global commands like send-mail do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as
`https://192.168.25.42/mgmt/tm/analytics/application-security/report`, along with a JSON body that specifies the name of the command.

Send an e-mail to recipients by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To send e-mail using the send-mail command, make a POST request. In the JSON body, specify the command. Specify the options, as well as the recipients, in the JSON body. Several of the property names use a hyphenated name instead of the camel case naming convention of iControl® REST.

```
POST /mgmt/tm/analytics/application-security/report
```

```
{
    "command":"send-mail",
    "view-by":"ip",
    "format":"pdf",
    "email-addresses":[
        "wchen@f5.com"
    ],
    "measures":[
        "illegal-transactions"
    ],
    "limit":20,
    "order-by":[
        {
            "measure":"illegal-transactions",
            "sort-type":"desc"
```

```
        }
    ],
    "smtp-config-override":"smtpserver"
}
```

Using the start command

Global commands like start do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as
`https://192.168.25.42/mgmt/tm/sys/icall/handler/perpetual`, along with a JSON body that specifies the name of the command.

Start a service by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To start a service using the start command, make a POST request. In the JSON body, specify the command and the name of the service.

```
POST /mgmt/tm/sys/icall/handler/perpetual
```

```
{
    "command":"start",
    "name":"perphd1"
}
```

Using the stop command

Global commands like stop do not have a direct mapping to an HTTP method, so you must use the POST method and specify an absolute URI, such as
`https://192.168.25.42/mgmt/tm/sys/icall/handler/perpetual`, along with a JSON body that specifies the name of the command.

Stop a service by making an iControl® REST request with the POST method and specifying the properties in a JSON body.

To stop a service using the stop command, make a POST request. In the JSON body, specify the command and the name of the service.

```
POST /mgmt/tm/sys/icall/handler/perpetual
```

```
{
    "command":"stop",
    "name":"perphd1"
}
```

Chapter

7

Application Security Manager

- *About differences in Application Security Manager (asm)*

About differences in Application Security Manager (asm)

Application Security Manager (asm) differs somewhat in implementation from the remainder of iControl® REST. The namespace for asm includes endpoints that are consistent with the iControl® REST namespace mgmt/tm/asm. As with any other organizing collection in iControl® REST, the structure of asm allows for discovery of the API. The set of error codes are similar to the iCR set of error codes. Tasks are supported in asm but transactions are not supported. Special endpoints like /example or /stats are not implemented in this release.

A URI in an asm namespace uses an MD5 hash as an identifier for resources.Open Data Protocol (OData) query parameters denote one area where asm differs from iControl® REST by providing a more complete implementation of $filter, $orderby, $select, $top, $skip, and $expand. However, asm does not implement any custom query parameters, such as expandSubcollections, or suffixes like /example or /stats in this release.

The implementation of asm supports the following HTTP methods:

Method	Description
GET	For both collections and resources, asm supports the GET operation to retrieve or search. The $filter query parameter support in asm includes more options than iControl®REST.
POST	For both collections and resources, asm supports the POST operation to create an entity. A POST request must include a JSON body, even if the body is empty.
DELETE	For collections, asm does not support the DELETE operation. For resources, asm supports the DELETE operation. By using a query parameter, asm also supports the option to delete multiple entities.
PUT	For collections, iControl REST does not support the PUT operation. For resources, iControl REST partially supports the PUT operation.
PATCH	For collections, asm does not support the PATCH operation. For resources, asm supports the PATCH operation. The PATCH method updates specified properties but does not reset or overwrite other properties. In asm, PATCH can update many entities by specifying a query option in the URI.
Other	For verbs that do not map to HTTP methods, the commands are specified in the JSON body of a POST request.

The following table lists the query parameters that are supported by asm.

Parameter	Description
$filter	Specifies a filter for a retrieve, update, or delete operation. In asm, this parameter implements the OData features.
$select	Specifies a subset of the properties to appear in the result set.
$skip	Specifies the number of rows to skip in the result set. The result set is chosen from the remaining rows.
$top	Specifies the first n rows of the result set.

As with iControl®REST, asm supports comparison and logical operators as described by the OData protocol.

Operator	Description
eq	Equal to

Operator	Description
ne	Not equal to
lt	Less than
le	Less than or equal to
gt	Greater than
ge	Greater than or equal to
and	True if both operands are true
or	True if either operand is true. In asm, $filter does not support this operator.
not	Negation of operand

The namespaces for `asm` are shown in the following table:

Namespace	Description
/tm/asm/attack-types	Collection, read-only.
/tm/asm/signatures	Collection that does not support update many or delete many requests.
/tm/asm/signature-statuses	Collection, read-only.
/tm/asm/signature-sets	Collection that does not support update many or delete many requests.
/tm/asm/signatures-update	Element
/tm/asm/signature-systems	Collection, read-only.
/tm/asm/policy-templates	Collection, read-only.
/tm/asm/policies	Collection that does not support update many or delete many requests. Collections within this namespace: • /tm/asm/policies/<MD5Hash>/methods • /tm/asm/policies/<MD5Hash>/filetypes • /tm/asm/policies/<MD5Hash>/cookies • /tm/asm/policies/<MD5Hash>/host-names • /tm/asm/policies/<MD5Hash>/blocking-settings/violations • /tm/asm/policies/<MD5Hash>/blocking-settings/evasions • /tm/asm/policies/<MD5Hash>/blocking-settings/http-protocols • /tm/asm/policies/<MD5Hash>/blocking-settings/web-services-securities • /tm/asm/policies/<MD5Hash>/urls • /tm/asm/policies/<MD5Hash>/parameters • /tm/asm/policies/<MD5Hash>/urls/<MD5Hash>/parameters • /tm/asm/policies/<MD5Hash>/whitelist-ips • /tm/asm/policies/<MD5Hash>/gwt-profiles • /tm/asm/policies/<MD5Hash>/json-profiles • /tm/asm/policies/<MD5Hash>/xml-profiles • /tm/asm/policies/<MD5Hash>/signatures • /tm/asm/policies/<MD5Hash>/signatures-sets

Tasks in asm support long-running asynchronous calls. Use the POST method to create a task and the GET method to query the status of the task. When you create a task, the response includes a `selfLink` for the task. To query for the status of a task, use that `selfLink` as a URI for a GET request. Similar to transactions,

a task has a status associated with it. The status values are NEW, STARTED, COMPLETED, and FAILURE. The namespaces for tasks are shown in the table.

Namespace	Description
/tm/asm/tasks/import-policy	Import a new policy, or replace an existing policy.
/tm/asm/tasks/export-policy	Export a policy, as XML.
/tm/asm/tasks/apply-policy	Apply changes to a policy.
/tm/asm/tasks/check-signatures	Check for the existence of a new signature file for download.
/tm/asm/tasks/update-signatures	Automatically download or manually set signatures.
/tm/asm/tasks/export-signatures	Export signatures.

Retrieving asm resources

Consistent with iControl® REST, the namespace for asm includes endpoints within the namespace /mgmt/tm/asm. As with any other organizing collection in iControl® REST, you can make a GET request to discover the resources of asm.

Use curl or a similar tool to make a request to the namespace /mgmt/tm/asm.

To discover the resources of asm, make a GET request to the root namespace to get the references, as shown in this example.

```
GET https://192.168.25.42/mgmt/tm/asm
```

```
{
    "selfLink":"https://localhost/mgmt/tm/asm",
    "kind":"tm:asm:asmcollectionstate",
    "items":[
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/asm/tasks"
            }
        },
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/asm/signature-update"
            }
        },
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/asm/policies"
            }
        },
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/asm/policy-templates"
            }
        },
        {
            "reference":{
                "link":"https://localhost/mgmt/tm/asm/signatures"
            }
```

```
      },
      {
         "reference":{
            "link":"https://localhost/mgmt/tm/asm/signature-statuses"
         }
      },
      {
         "reference":{
            "link":"https://localhost/mgmt/tm/asm/signature-sets"
         }
      },
      {
         "reference":{
            "link":"https://localhost/mgmt/tm/asm/signature-systems"
         }
      },
      {
         "reference":{
            "link":"https://localhost/mgmt/tm/asm/attack-types"
         }
      }
   ]
}
```

This example expands one of the links in the response from the previous request by making another GET request for a resource. Take note that each URI contains hash strings as resource identifiers.

```
GET https://192.168.25.42/mgmt/tm/asm/policies
```

```
{
   "selfLink":"https://localhost/mgmt/tm/asm/policies",
   "kind":"tm:asm:policies:policycollectionstate",
   "items":[
      {
         "policyBuilderReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/policy-builder"

         },
         "blockingSettingReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/blocking-settings",

            "isSubCollection":true
         },
         "cookieReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/cookies",
            "isSubCollection":true
         },
         "hostNameReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/host-names",
            "isSubCollection":true
         },
          "selfLink":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A",

         "stagingSettings":{
            "signatureStaging":true,
            "enforcementReadinessPeriod":7
         },
         "versionDeviceName":"10000-1-E12U39.sh",
         "signatureReference":{
```

```
"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/signatures",
          "isSubCollection":true
        },
        "createdDatetime":"2013-12-06T19:29:54Z",
        "filetypeReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/filetypes",
          "isSubCollection":true
        },
        "id":"MwavowFbOsSD-Fgt4trP6A",
        "modifierName":"admin",
        "versionDatetime":"2013-12-26T23:12:57Z",
        "subPath":"/Common",
        "versionLastChange":"Policy Attributes  [update]: Policy Builder
determined that security policy \"/Common/my-VS\" is unstable.",
        "active":true,
        "caseInsensitive":false,
        "name":"my-VS",
        "description":"",
        "fullPath":"/Common/my-VS",
        "policyBuilderEnabled":true,
        "trustXff":false,
        "partition":"Common",
        "attributes":{
            "pathParameterHandling":"as-parameters",
            "triggerAsmIruleEvent":"disabled",
            "maskCreditCardNumbersInRequest":true,
            "inspectHttpUploads":false,
            "maximumHttpHeaderLength":2048,
            "maximumCookieHeaderLength":2048,
            "useDynamicSessionIdInUrl":false
        },
        "xmlProfileReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/xml-profiles",

          "isSubCollection":true
        },
        "methodReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/methods",
          "isSubCollection":true
        },
        "customXffHeaders":[

        ],
        "creatorName":"admin",
        "kind":"tm:asm:policies:policystate",
        "urlReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/urls",
          "isSubCollection":true
        },
        "virtualServers":[
            "/Common/my-VS"
        ],
        "headerReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/headers",
          "isSubCollection":true
        },
        "protocolIndependent":false,
        "lastUpdateMicros":1.386358822e+15,
        "signatureSetReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/signature-sets",

          "isSubCollection":true
```

```
        },
        "allowedResponseCodes":[
            400,
            401,
            404,
            407,
            417,
            503
        ],
        "parameterReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/parameters",
            "isSubCollection":true
        },
        "jsonProfileReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/json-profiles",

            "isSubCollection":true
        },
        "applicationLanguage":"utf-8",
        "enforcementMode":"transparent",
        "isModified":false,
        "gwtProfileReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/gwt-profiles",

            "isSubCollection":true
        },
        "whitelistIpReference":{

"link":"https://../mgmt/tm/asm/policies/MwavowFbOsSD-Fgt4trP6A/whitelist-ips",

            "isSubCollection":true
        },
        "versionPolicyName":"/Common/Dummy-VS"
    }
  ]
}
```

To use a GET request to search for matches, use a query parameter in the URI, as shown in the following example.

```
GET https://192.168.25.42/mgmt/tm/asm/policies?$filter=name eq my-VS
```

Note: The current release does not support the use of the `or` *operator in a $filter query.*

Creating asm resources

Consistent with iControl® REST, the namespace for `asm` includes endpoints within the namespace `/mgmt/tm/asm`. As with any other organizing collection in iControl® REST, you can make a POST request to create resources in `asm`.

Create a new resource by making a POST request to a namespace `/mgmt/tm/asm`..

Updating asm resources

Consistent with iControl® REST, the namespace for asm includes endpoints within the namespace /mgmt/tm/asm. As with any other resources in iControl® REST, you can make a PATCH request to update resources in asm.

Update a resource by making a PATCH request to a resource in the namespace /mgmt/tm/asm..

To update resources in asm, make a PATCH request, as shown in this example. The JSON body is shown in the second block.

```
PATCH https://192.168.25.42/mgmt/tm/asm/policies/<MD5HASH>/urls/
```

```
{
    "clickjackingProtection": true,
    "clickjackingtype":"Never"
}
```

To use a PATCH request to update multiple entities, use a query parameter in the URI. The JSON body is shown in the second block.

```
PATCH https://192.168.25.42/mgmt/tm/asm/policies/<MD5HASH>/urls?$filter=type
 eq explicit
```

```
{ "staging": false }
```

Deleting asm resources

Consistent with iControl® REST, the namespace for asm includes endpoints within the namespace /mgmt/tm/asm. As with any other resources in iControl REST, you can make a DELETE request to delete resources in asm.

Delete a resource by making a DELETE request to a resource in the namespace /mgmt/tm/asm.

To delete resources in asm, make a DELETE request, as shown in this example. The response is shown as follows. For asm, the response contains the deleted resource.

```
DELETE
https://192.168.25.42/mgmt/tm/asm/tasks/import-policy/ZuJ5QPuFj9r_LwbrDgoPsg
```

```
{
    "isBase64":false,
    "status":"FAILURE",
    "name":"TCB policy",
    "lastUpdateMicros":1.389135008e+15,
    "kind":"tm:asm:tasks:import-policy:import-policy-taskstate",
```

```
"selfLink":"https://../mgmt/tm/asm/tasks/import-policy/ZuJ5QPuFj9r_LwbrDgoPsg",

    "filename":"tcbpolicy.xml",
    "id":"ZuJ5QPuFj9r_LwbrDgoPsg",
    "startTime":"2014-01-07T22:50:08Z",
    "result":{
        "message":"Exported policy file not found!."
    }
}
```

To use a DELETE request to delete multiple entities, use a query parameter in the URI, as shown in this example.

```
DELETE https://192.168.25.42/mgmt/tm/asm/policies/<MD5HASH>/urls/?$filter=staging
eq true
```

Chapter
8

Additional features

- *About HTTP response codes*
- *About log files*
- *About public URIs*
- *List of public URIs*

About HTTP response codes

Responses to all iControl® REST requests contain a response code, as listed here.

Success responses

Response code	Description
200 OK	Indicates success for all methods.

Error responses

Response code	HTTP methods	Description
400 Bad Request	all	Possible causes include: • malformed HTTP request • incorrect name for a resource in a request
401 Unauthorized	all	Possible causes include: • missing HTTP authorization header • insufficient permissions for the credentials supplied for an administrator
403 Forbidden	all	Possible causes include: • insufficient permissions for the credentials supplied for an administrator • attempt to perform an unsupported action, such as deleting a property
404 Not Found	all	Possible causes include: • attempting to access a resource that no longer exists in the database
409 Conflict	POST, PUT	Possible causes include: • attempting to create a resource that already exists Indicates a conflict between the requested state change and the current state of the resource. For example, this is the error response if you POST a resource that already exists.
415 Unsupported Media Type	POST, PUT	Possible causes include: • specifying an incorrect `Content-Type` header value • specifying a malformed JSON body with a POST or PUT request
500 Internal Server Error	all	Possible causes include: • attempting to access iControl REST when the process is not running

Response code	HTTP methods	Description
501 Not Implemented	POST	Possible causes include: • attempting to access a endpoint that does not exist • attempting to invoke an unsupported `tmsh` command through iControl REST

About log files

From the console or an SSH connection to your BIG-IP® device, you can find the following log files for iControl® REST:

- `/var/log/restjavad-audit.0.log` shows all authentications to the iControl REST service. This is an ordered list of every REST call.
- `/var/log/restjavad.0.log` contains information about connections to the iControl REST service, such as errors returned.
- `/var/log/icrd` shows the actions of the `icrd` process, which manages the threads for processing the REST calls.
- `/var/log/ltm` contains messages from `mcpd`, a process called by `icrd` that manages the system configuration.

Use standard `Unix` commands to work with these files, such as `tail`, `grep`, and `less`. In this example, the session logs in to a BIG-IP system through `ssh` and uses `tail -f` to monitor the `/var/log/restjavad-audit.0.log` log file:

```
juser@bench2:~/$ ssh root@192.168.25.42
Password: default
Last login: Fri Mar 29 09:03:25 2013 from 192.168.98.174
[root@localhost:Active:Standalone] config # tail -f
/var/log/restjavad-audit.0.log
[I][339][29 Mar 2013 16:04:06 UTC][ForwarderPassThroughWorker] \
  [run]{"user":"admin","method":"PUT",\
  "uri":"http://localhost:8100/mgmt/tm/ltm/pool/dns-pool2",\
  "status":"succeeded","from":"192.168.96.37"}
[I][340][29 Mar 2013 16:04:06 UTC][ForwarderPassThroughWorker] \
  [run]{"user":"admin","method":"GET",\
  "uri":"http://localhost:8100/mgmt/tm/ltm/pool","\
  status":"succeeded","from":"192.168.96.37"}
[I][341][29 Mar 2013 16:04:06 UTC][ForwarderPassThroughWorker] \
  [run]{"user":"admin","method":"DELETE",\
  "uri":"http://localhost:8100/mgmt/tm/ltm/pool/test-pool2",\
  "status":"succeeded","from":"192.168.96.37"}
[I][342][29 Mar 2013 16:04:07 UTC][ForwarderPassThroughWorker] \
  [run]{"user":"admin","method":"POST",\
  "uri":"http://localhost:8100/mgmt/tm/sys/folder",\
  "status":"succeeded","from":"192.168.96.37"}
[I][343][29 Mar 2013 16:04:07 UTC][ForwarderPassThroughWorker]\
  [run] {"user":"admin","method":"DELETE",\
  "uri":"http://localhost:8100/mgmt/tm/sys/folder/~fw_objs",\
  "status":"succeeded","from":"192.168.96.37"}
[I][344][29 Mar 2013 16:04:07 UTC][ForwarderPassThroughWorker]\
  [run] {"user":"admin","method":"DELETE",\
  "uri":"http://localhost:8100/mgmt/tm/sys/folder/~eu~east~romania",\
  "status":"succeeded","from":"192.168.96.37"}
[I][345][29 Mar 2013 16:04:07 UTC][ForwarderPassThroughWorker]\
```

```
  [run] {"user":"admin","method":"POST",\
  "uri":"http://localhost:8100/mgmt/shared/authz",\
  "status":"succeeded","from":"192.168.96.37"}
[I][346][29 Mar 2013 16:04:07 UTC][ForwarderPassThroughWorker]\
  [run] {"user":"admin","method":"GET",\
  "uri":"http://localhost:8100/mgmt/shared/authz",\
  "status":"succeeded","from":"192.168.96.37"}
[I][347][29 Mar 2013 16:04:10 UTC][ForwarderPassThroughWorker]\
  [run] {"user":"dns_admin","method":"GET",\
  "uri":"http://localhost:8100/mgmt/tm/sys",\
  "status":"succeeded","from":"192.168.96.37"}
[I][350][29 Mar 2013 16:04:10 UTC][ForwarderPassThroughWorker]\
  [run] {"user":"admin","method":"GET",\
  "uri":"http://localhost:8100/mgmt/tm/ltm/pool/http-pool?$stats=true",\
  "status":"succeeded","from":"192.168.96.37"}
...
```

If you need to adjust the logging levels for `icrd`, contact F5® Networks Technical Support (`http://www.f5.com/support/`).

About public URIs

A URI is considered to be public if you can access it through an iControl® REST request. In general, all of the following are public:

- Traffic Management Shell (tmsh) modules
- Traffic Management Shell (tmsh) components
- Any component properties that are accessible through the `tmsh show` command.

To view the component properties, make a GET request of a parent component. By default, you cannot use a GET request to obtain them directly through a public URI.

The public URIs exist to provide direct access to some of those component properties. The iControl REST process allows these for convenience, for situations where a PUT request of the entire containing object (a component or collection) would be unwieldy.

In many cases, the second-to-last part of the path is the name of a component, and you need to provide a specific object name for that component before the final part of the path. For example, to access the public URI `/mgmt/tm/gtm/pool/members`, you must specify the GTM™ pool for which you want members, such as `/mgmt/tm/gtm/pool/pool5/members` for the members of `pool5`.

List of public URIs

iControl® REST contains these public URIs:

- `/mgmt/tm/apm/aaa/oam/accessgates`
- `/mgmt/tm/apm/aaa/securid/config-files`
- `/mgmt/tm/apm/policy/agent/endpoint-check-software/items`
- `/mgmt/tm/apm/policy/customization-group/templates`
- `/mgmt/tm/apm/profile/access/domain-groups`
- `/mgmt/tm/apm/profile/connectivity/client-policy`
- `/mgmt/tm/apm/resource/app-tunnel/apps`

- /mgmt/tm/apm/resource/client-traffic-classifier/entries
- /mgmt/tm/apm/resource/network-access/optimized-app
- /mgmt/tm/apm/resource/portal-access/items
- /mgmt/tm/apm/resource/sandbox/files
- /mgmt/tm/apm/sso/form-basedv2/forms
- /mgmt/tm/auth/remote-role/role-info
- /mgmt/tm/gtm/pool/members
- /mgmt/tm/gtm/server/virtual-servers
- /mgmt/tm/ltm/dns/dnssec/key/generation
- /mgmt/tm/ltm/policy-strategy/operands
- /mgmt/tm/ltm/policy/rules
- /mgmt/tm/ltm/policy/rules/actions
- /mgmt/tm/ltm/policy/rules/conditions
- /mgmt/tm/ltm/pool/members
- /mgmt/tm/ltm/profile/analytics/alerts
- /mgmt/tm/ltm/profile/analytics/traffic-capture
- /mgmt/tm/ltm/profile/rewrite/uri-rules
- /mgmt/tm/ltm/virtual/fw-rules
- /mgmt/tm/net/route-domain/fw-rules
- /mgmt/tm/net/router-advertisement/prefixes
- /mgmt/tm/net/self/fw-rules
- /mgmt/tm/net/wccp/services
- /mgmt/tm/pem/policy/rules
- /mgmt/tm/pem/policy/rules/flow-info-filters
- /mgmt/tm/security/dos/profile/application
- /mgmt/tm/security/dos/profile/protocol-dns
- /mgmt/tm/security/dos/profile/protocol-sip
- /mgmt/tm/security/firewall/global-rules/rules
- /mgmt/tm/security/firewall/management-ip-rules/rules
- /mgmt/tm/security/firewall/policy/rules
- /mgmt/tm/security/firewall/rule-list/rules
- /mgmt/tm/security/log/profile/application
- /mgmt/tm/security/log/profile/network
- /mgmt/tm/security/log/profile/protocol-dns
- /mgmt/tm/security/log/profile/protocol-sip
- /mgmt/tm/sys/application/template/actions
- /mgmt/tm/sys/file/apache-ssl-cert/bundle-certificates
- /mgmt/tm/sys/file/ssl-cert/bundle-certificates
- /mgmt/tm/sys/file/system-ssl-cert/bundle-certificates
- /mgmt/tm/sys/icall/handler/perpetual/subscriptions
- /mgmt/tm/sys/icall/handler/triggered/subscriptions
- /mgmt/tm/sys/ltcfg-class/fields
- /mgmt/tm/sys/ltcfg-instance/fields
- /mgmt/tm/sys/ntp/restrict
- /mgmt/tm/sys/snmp/communities
- /mgmt/tm/sys/snmp/traps
- /mgmt/tm/sys/snmp/users
- /mgmt/tm/wam/policy/nodes

Additional features

- `/mgmt/tm/wam/policy/nodes/invalidations`
- `/mgmt/tm/wam/policy/nodes/invalidations/cache-content`
- `/mgmt/tm/wam/policy/nodes/invalidations/request`
- `/mgmt/tm/wam/policy/nodes/matching`
- `/mgmt/tm/wam/policy/nodes/proxy`
- `/mgmt/tm/wam/policy/nodes/proxy-override`
- `/mgmt/tm/wam/policy/nodes/substitutions`
- `/mgmt/tm/wam/policy/nodes/variation`

Index